Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Quiz 2E **Spring 2020, CU-Boulder**

Instructions: This quiz is closed book, closed book, and an individual effort. Electronic devices (including, but not limited to phones, smart watches, FitBits, calculators, etc.) are NOT allowed on your person (including in your pocket). Possession of such electronics is grounds to receive a 0 on this quiz. Proofs should be written in **complete sentences. Show all work to receive full credit.**

Please provide these:

Left neighbor name :

Right neighbor name :

Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**

Quiz 2E **Spring 2020, CU-Boulder**

1. ***Standard 2.*** *Let $f(n) = n^2 + 2n + 8$ and let $g(n) = 4^n$. Determine which relation* ***best*** *applies: $f(n) \leq \mathcal{O}(g(n))$, $f(n) \geq \Omega(g(n))$, or $f(n) = \Theta(g(n))$. Clearly show* ***ALL*** *your work.*

   Solutions: We apply the Limit Comparison Test. We have that:

$$\begin{aligned} \lim_{n\to\infty} \frac{n^2+2n+8}{4^n} &= \lim_{n\to\infty} \frac{2n+2}{\ln(4)4^n} && (1) \\ &= \lim_{n\to\infty} \frac{2}{(\ln(4))^2 4^n} && (2) \\ &= 0. && (3) \end{aligned}$$

   We note that (1) and (2) follow from L'Hopital's rule, where we have the indeterminate form $\frac{\infty}{\infty}$. As:

$$\lim_{n\to\infty} \frac{n^2+2n+8}{4^n} = 0,$$

   we have that $f(n) \leq \mathcal{O}(g(n))$, but $f(n) \neq \Theta(g(n))$.

   Remark: A common error we saw was that students indicated the derivative of 4^n was $n4^{n-1}$. This is **not** correct, as 4^n is an exponential, not a polynomial. The derivative of 4^n is $\ln(4)4^n$.

Name:

ID:

CSCI 3104, Algorithms **Profs. Chen & Grochow**
Quiz 2E **Spring 2020, CU-Boulder**

2. **Standard 3.** *Analyze the worst-case runtime of the following algorithms. You should give your answer in big-Oh notation; try to give as tight a bound as possible (but you* do not *need to give an input which achieves your worst-case bound).*

   - *In the column to the right of the code, indicate the cost of each line*
   - *In the next column (all the way to the right), indicate the number of times each line is executed*
   - *Below the code, justify your answers (show your work), and compute the total runtime in terms of big-Oh notation.*
   - *In both columns, you don't have to put the* exact *values. For example, putting "c" for constant is fine, and the difference between n and $n-1$ is irrelevant here.*

```
                                                 | Cost | # times run
// A is a square, 2D array; indexed starting from 1
// len(A) = n
1  f(A[1, ..., n][1, ..., n]):                   |      |
2    let d be a copy of A                        | n^{2} | 1
3    for i = 1 to len(A):                        | 1     | n
4      d[i][i] = 0                               | 1     | n
5
6    for (i = 1; i < len(A); i = i * 2):         | 1     | \log_{2}(n)
7      for j = 1 to len(A):                      | 1     | n
8        for k = 1 to len(A):                    | 1     | n
9          if (d[i][k] + d[k][j]) < d[i][j]:     | 1     | n
10           d[i][j] = d[i][k] + d[k][j]         | 1     | n
11
12   return d                                    | 1     | 1
```

Solution: *We have that:*

$$T(n) = n^2 + n + \log_2(n) \cdot n^2 + 1 = \Theta(n^2 \log_2(n)).$$

The first n^2 here is from line 2. The next n is from lines 3–4. Since the three loops on lines 6–8 are independent of one another, the number of times lines 9–10 run is the product of the number of times the three loops run, which is $n^2 \log_2 n$.

Remark: *Students received full credit if they specified that line 2 in the algorithm took a single step, rather than n^2 steps.*